8/5/1999
Version 1.0

Draft

DFAS Oracle Designer
Design Transformer Standards and Guidlines

August 5, 1999

8/5/1999
Version 1.0

Draft DFAS Oracle Designer Design Transformer Standards and Guidelines

Executive Summary

The purpose of the DFAS Oracle Designer Design Transformer Standards and Guidelines is to provide detailed guidance for DFAS projects that will be using Oracle Designer. Included in this document is detailed information on the settings for the design transformers. Examples are guidelines for setting the usage of prefixes or module implementation granularity.

Topics covered include database design transformer and application design transformer standards and guidelines.

For additional information please reference the DFAS Oracle Designer Design Standards and Guidelines. This paper addresses the specific naming conventions and tool rules for DCII applications. The Transformer Standards and Guidelines should be used in parallel with the Design Standards and Guidelines.

Database Design Transformer

STANDARDS

DDT-001: The database name selected is “<none>”. This means that the database and tablespace settings in Designer may be left blank. The table definitions are initially generated without defining/implementing against a database, tablespace or storage clause. This is also true for the generation of the index definition under the table definition.

DDT-002: The cascade rules for the newly generated foreign key definitions should be the default value of “RESTRICTED”. Cascade rules will depend on business rules and should be handled on a case by case basis if the default is not used. Document any deviations from using RESTRICTED in the entity model description.

DDT-003: The surrogate keys are to use the shared sequence domains. These domains are of the format “SEQ#”, where the # is the length of the key.

DDT-004: The maximum identifier length should be the default value of 30.

DDT-005: Prefixes are to be generated for foreign key columns. The prefix is the table alias.

DDT-006: Prefixes will not be generated for surrogate key columns.

DDT-007: Columns will not be prefixed.

DDT-008: The prefix for the table will be a 2- or 3-character application acronym followed by an underscore (“_”). Common tables belonging to the DCD will have the prefix of “DC_”.

DDT-009: The ordering of the columns should be as follows:

1. Primary Key Columns
2. Mandatory Columns
3. Discriminator Columns
4. Foreign Key Columns before Attribute Columns
5. Grouping Columns by their Source Entity

NOTE: Long and long raw columns should be moved to the bottom of the column list after the table is generated. Currently only one of these types of columns is allowed per table by the Oracle database.

GUIDELINES

DDT-010: The commit frequency allows the developer to determine how the results of the transformation being executed are automatically saved to the database. The selections should be used as follows:

"Don't commit" - A commit is not issued as part of normal processing. Used as a preliminary "test" of the transformation process to determine the correctness of the results. Developer may commit transformation work after reviewing results of the transformation activity.

"At end of run" - A commit is issued after the completion of the transformation. Used when the number of elements selected for transformation is relatively small. Also valuable when incrementally executing the transformation; works similar to "after each phase" selection.

"After each phase" - commit points are created as each element type, Primary or Secondary, is processed through the various transformation step modes selected.. Used when processing a significant number of elements and by default.

Steps:
Determine the types of elements to create or modify
Execute the "transformation" in a test mode ("Don't commit")
Review transformation report contents
Consider any changes required - changes to settings options, subtype resolution, prefix creation, …
Execute the "transformation" in final mode ("After each phase" or "At end

of run")

Note: Selection "At end of run" and "Don't commit" may require a larger rollback segment depending on the number of elements selected and the amount of work performed.

DDT-011: The “Create surrogate keys for all new tables” check box should be marked whenever it is desired to let the Designer tool create surrogate keys. It is recommended that the generation of tables needing the creation of surrogate keys be separate from the generation of tables that already have a primary unique identifier defined.

DDT-012: Constraints will almost always be implemented on the server and will usually be implemented on the client. Therefore the selection for where to implement constraints will usually be BOTH the server and the client. Business rules will determine where the constraints need to be implemented.

DDT-013: The “Allow instantiable super-types” check box will be checked only when the super-type needs a discriminator value distinct from the sub-type values.

Application Design Transformer Guidelines

The ADT is a repository utility that works like the DDT; it creates Design elements from Analysis elements. In creates modules from functions. The ADT should be run after the low-level functions have entity and attribute usages completely defined. The function model should be as complete as possible before you run ADT, so it will give you a set of modules that closely matches your needs. ADT should be run with two different settings for each function branch; first to create candidate modules (generated modules you need to explicitly accept) from functions and second to create a module structure from the function business unit usages.

ADT-001: Generate Options: Select Candidate modules for menu generation - you cannot generate modules that are menus unless you have modules that have been accepted.

ADT-002 Common Parameters: This is where the start function is designated. ADT generates a module for this and all other applicable functions under it in the function hierarchy. If you want to generate modules for all functions leave this field blank. The Module Prefix allows you to designate a module prefix that ADT will use when creating the module.
Application prefixes examples include:

- GET_
- COM_
- SGL_
- DSDS_

- CFT_ or CEFT_

Module type identifiers are:

- F = Form
- R = Report
- U = Function
- S = Shell
- M = Menu
- T = Triggers
- K = Package
- W = Web
- P = Procedure

Purpose identifiers examples include:

- M – modify
- P – populate

All modules will have a Short Name property value of this prefix followed by a four-digit number. ADT also inserts a value for the Name property of each generated module definition. The default for this property is the uppercase Short Definition property of the function, but you can modify the default by editing the value. You can also specify the number at which the module names will start. (Start Number) The Find Highest button fills in a number based on the highest module number already generated. You need to enter a number between 10 and 9999 before clicking this button.

ADT-003 Module Options: If you are running the ADT for modules, the Module Options area is enabled so you can indicate the language for screen, report, and utility modules that ADT generates. When creating candidate module ADT creates a module type. The type is assigned as a module property along with the settings that you specify here. The default settings are Developer Forms, Developer Reports, and PL/SQL. The Language area drop-down lists for Screen, Report and Utility include other options besides the supported generators. Specify Forms, Reports, Visual Basic, or Web Server if you will be using one of these generators.

ADT-004 Merge Granularity: When creating modules, you can determine the rule ADT uses to group functions into modules. You choose one of three options for Merge Granularity:

Identical Entities merges functions into one module if they have the same entities mapped to them.

Identical Entities and Usages combine functions into one module if the functions have the same entities and same CRUD for those entities.

Identical Attributes joins functions into one module if those functions have the same entities and attributes mapped to them.

ADT-005 Menu Options: When generating menus in the ADT run, this area is enabled and the Module Options area is disabled. Here you choose the menu language (Developer Forms is the default). Max Options on Menu defines the maximum the number of items you can have in each menu. If there are more, ADT creates another menu to hold them. The last setting in this area is Include Manual Options. If you check this, ADT will add the modules it determined to be manual (those without data usages) into the menu structure. You can use these for information screens that tell the user what to do manually, if requirements dictate this need.

ADT-006 Unlike the Database Design Transformer, the Application Design Transformer is not intended to be run iteratively to modify definitions of previously created elements

ADT-007 If you need to start over with ADT but have created modules, you must delete the modules first or you will get naming conflicts errors when you run the utility again. If deleting a module in RON fails because the module is called by other modules, open the Usages of the module and record the "called by modules" and all passed values and arguments. Then use the RON menu bar select Utilities - Force Delete to remove the module from the repository.

ADT-008 Modules have dependencies that mandate a certain sequence for the tasks you perform in the design transformer (DDT). Be sure to run the Database Design Transformer to create tables definitions before running the Application Design Transformer. Also be sure to run the ADT to create menu modules.

ADT-009 If ADT merges functions into one module, it will also merge the text (in the Description and Note Property) from those functions.

ADT –010 Function Mapping Guidelines: In general one module is created for each function in the hierarchy under the one you specify as the Start Function when you run the utility. A function must be one of three types: a leaf (atomic) function with no elementary or common function parents, a common function, or an elementary function. A leaf function is at the end of the hierarchy tree (it has no children). A common function appears more than once in the function hierarchy. An elementary function is one for which you have set the Elementary property to Yes. ADT merges modules if they are not manual (that is, they have entity and attribute usages) and are associated with functions that have the same Response Needed property, input parameters business unit usages, and similar data usages (based on granularity you set in the ADT window). ADT will duplicate a module if the function it is based on had more than one business unit usage. ADT will create one module for each business unit for that function.

ADT-011 Module Categorization Guidelines: After creating modules ADT categorizes them into types. It categorizes a module as manual if no tables are implemented for entities of the function or if the function had no entities and attributes associated with it or if the function has entities but no attributes. It creates a report module if the function had read-only (retrieve) data usages. It produces a utility module if the Response Needed property of the corresponding function is Overnight. It creates a screen if the Response Needed property of the corresponding function is Immediate. If none of these conditions apply the module becomes a screen type.

ADT-012 Module Grouping Guidelines for Menus: The application-level menu module is generated as the top level (main menu), and this module calls the first-level menu modules. ADT generates the first level of menu modules from the module business unit usages (derived from the function business unit usages). If the module is not associated with any business unit, ADT groups it in a Miscellaneous menu. The menu system as the following structure which can be changed if needed:

Top Level	Application System (main menu)
First Level	Business unit grouping (top menu items)
Second Level	Module type grouping (pull-down menu items)
Third Level	Individual modules under the specific modules

ADT-013 Module Data Usage Guidelines - Each module that ADT creates will contain one module component and a corresponding table usage from entity usage in the function the module is based on. Column usages are similar; they are mapped to bound items in the table if the function has the corresponding attributes mapped into it. If DDT created a surrogate foreign key for a table, that key will be created in the module component usage even though there is no corresponding attribute. Another rule is that if a table has an INSERT usage on the module, ADT will add all mandatory columns as bound item's usages whether or not there are corresponding attribute usages.